Build Community Android Distribution and Ensure the Quality

Jim Huang (黃敬群) <**jserv**@0xlab.org>

Developer & Co-Founder, 0xlab

http://0xlab.org/

Oct 28, 2011 / ELC Europe (Prague)

Rights to copy

connect your device to application

0xlab

$0x1ab = 16^2+16x10+11 = 427$

(founded on April 27, 2009)

0xlab is another Hexspeak.

About Me

(1) Come from Taiwan

(2) Contributior of Android Open Source Project (AOSP)

(3) Developer, Linaro

(4) Contributed to GNU Classpath / Kaffe, Linux internationalization (i18n), Openmoko

Commercial Partners of 0xlab

ARM / Linaro

Contribute to Linaro Android since the first line of code

AzureWave

Build wireless networking & image processing solutions

Mediatek

Android based consumer products

Open Embedded Software Foundation

Contribute to the reference implementation

Eventually, partners can benefit from open source efforts and our experience.

http://0xlab.org/partners.html

Agenda

(1) Build Android distribution

(2) Lesson learned from AOSP

(3) Ensure the Quality

(4) Bring enhancements back to Community

Build Android Distribution

based on non typical open source projects

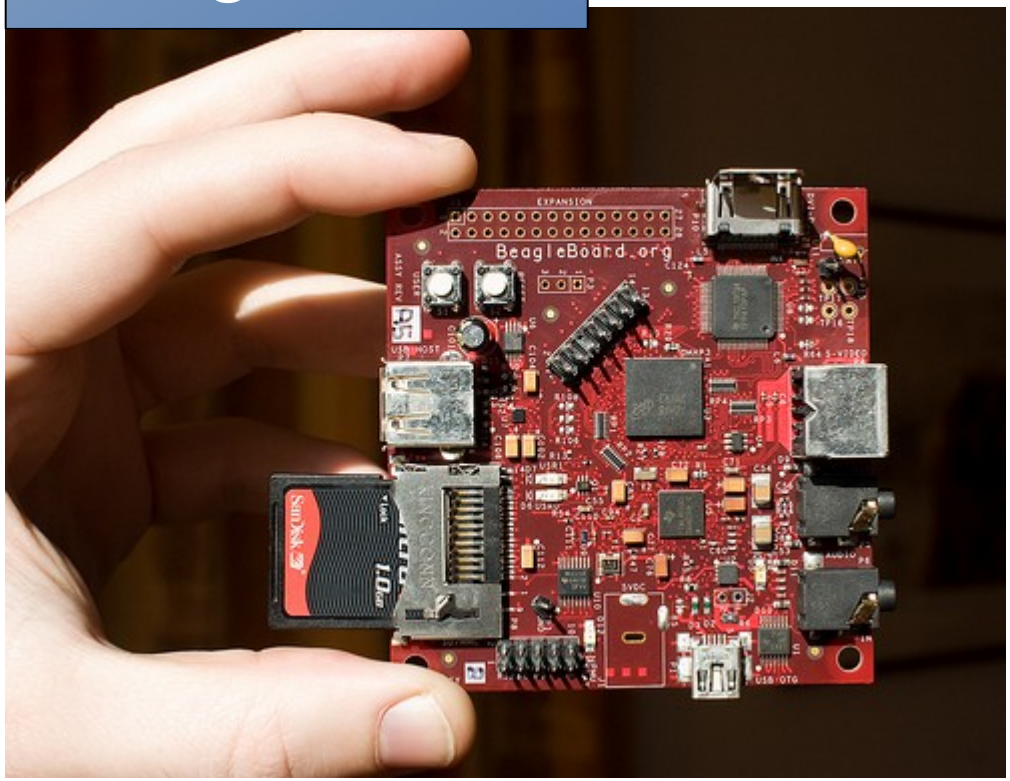
The reason why we built community Android Distribution:

Initially, we just wanted to enable wireless connectivity features on Android for our hardware partnters.

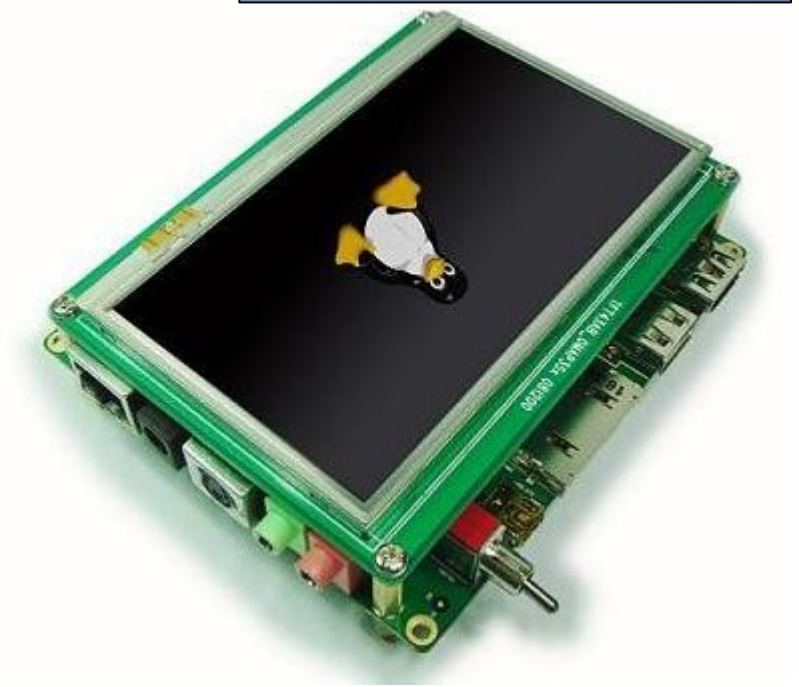
But, we never thought that it was difficult to do things efficiently.

0xdroid: enhanced Android distribution

Beagleboard



DevKit8000



We suffered from performance and usability issues in AOSP. 0xdroid is basically the environment where we can develop and experiment.

What does 0xdroid deliver?

- Hardware enablement: Beagleboard (TI OMAP3), Pandaboard (TI OMAP4), Snowball (ST-Ericsson Ux500; on-going)
- Provide full source code for HAL
- Usability: software cursor, window manager fix, large screen tweaks, network connectivity fix
- Performance: ARM specific optimizations, graphics enhancement
- Features: Bluetooth HID (keyboard/mouse), external modem, 3D effects, customized Launcher

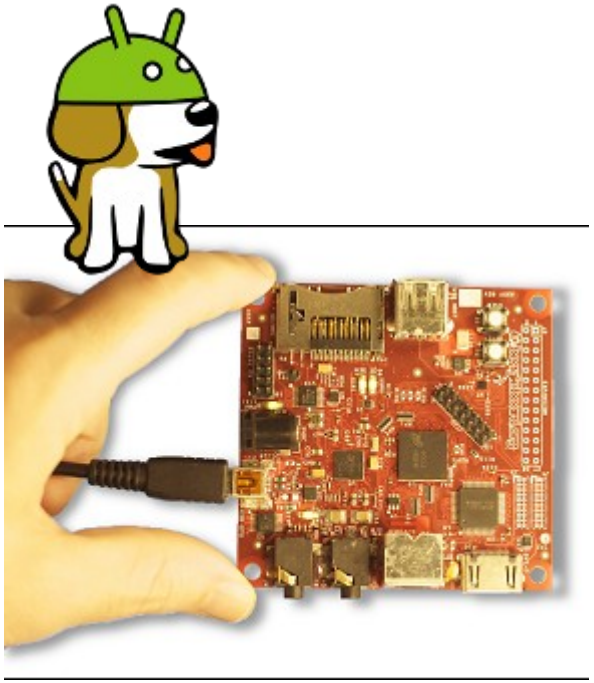
But, 0xlab is not really making yet another Android distribution. We wish to help community.

0xdroid is just a *testbed* (or reference implementation), and the valuable changes should be merged in upstream or other community projects.

Strategy and Policy

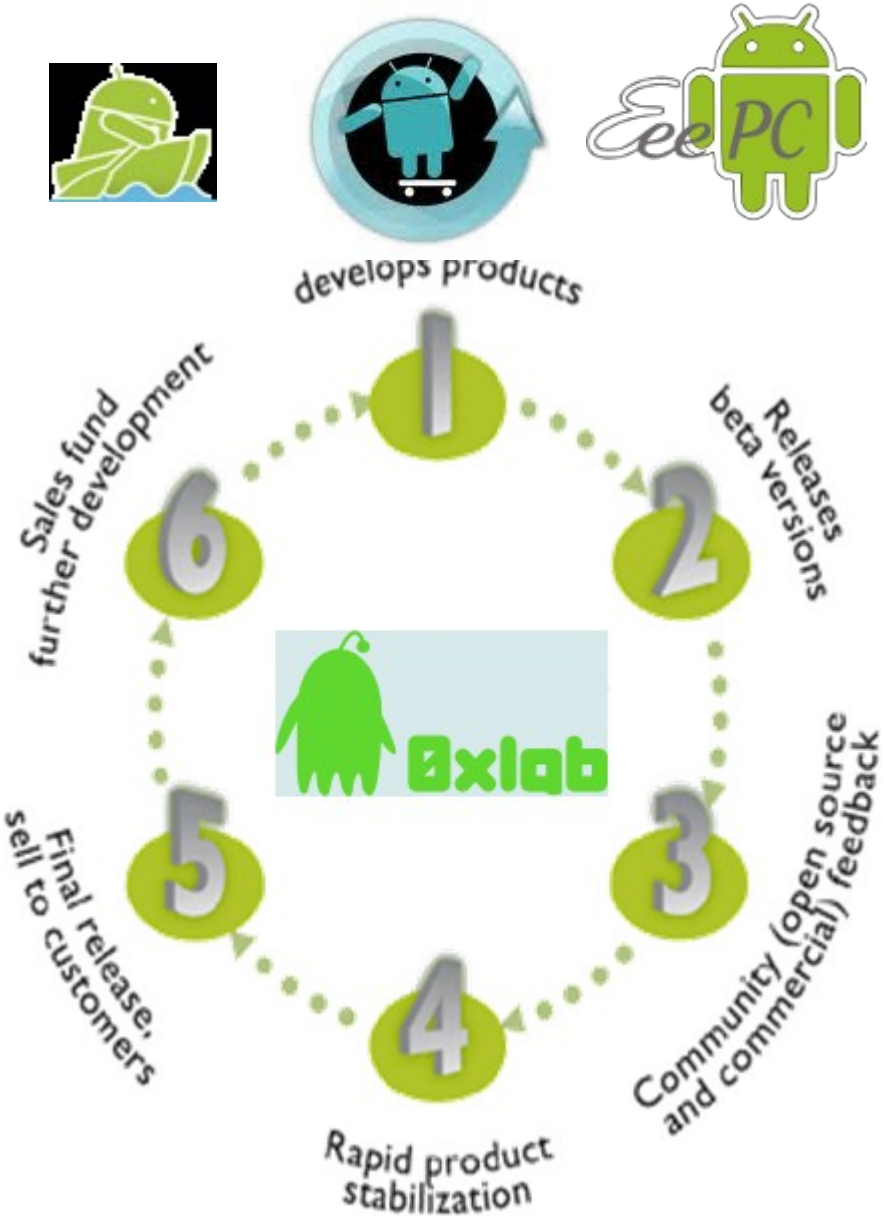
- open source efforts to improve AOSP
- We focus on small-but-important area of Android.
  - **toolchain, libc, dynamic linker, skia, software GL, system libraries, HAL, UX**
- Develop system utilities for Android
  - **benchmark, black-box testing tool, validation infrastructure**
- Feature driven development
  - **Faster boot/startup time, Bluetooth profile, visual enhancements**
- Submit and share changes to...
  - **AOSP, CyanogenMod, Android-x86, and Linaro**

Working Model by 0xlab



Linaro

Rowboat **CyanogenMod** **Android-x86**



Lesson Learned from AOSP

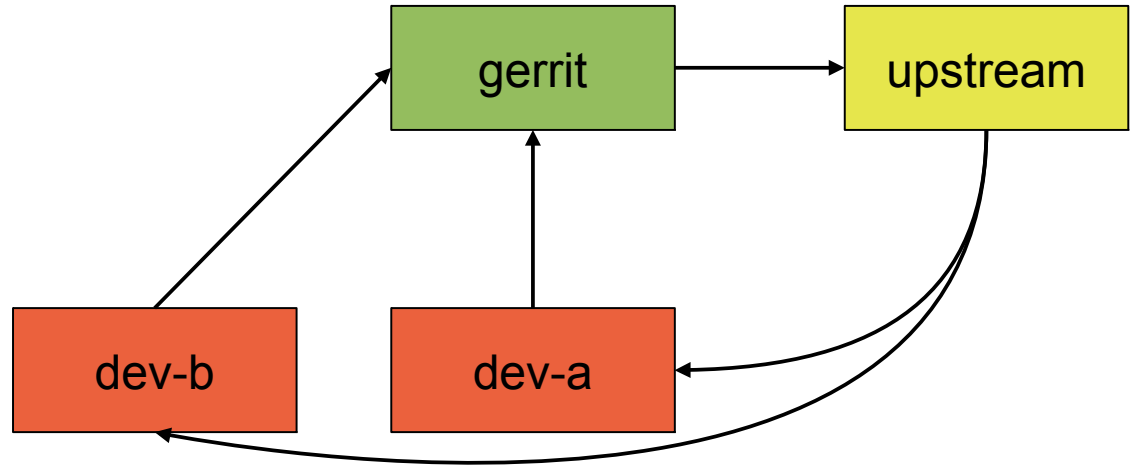
Let's go Upstream!
Unfortunately, contributing to AOSP is an __art__.

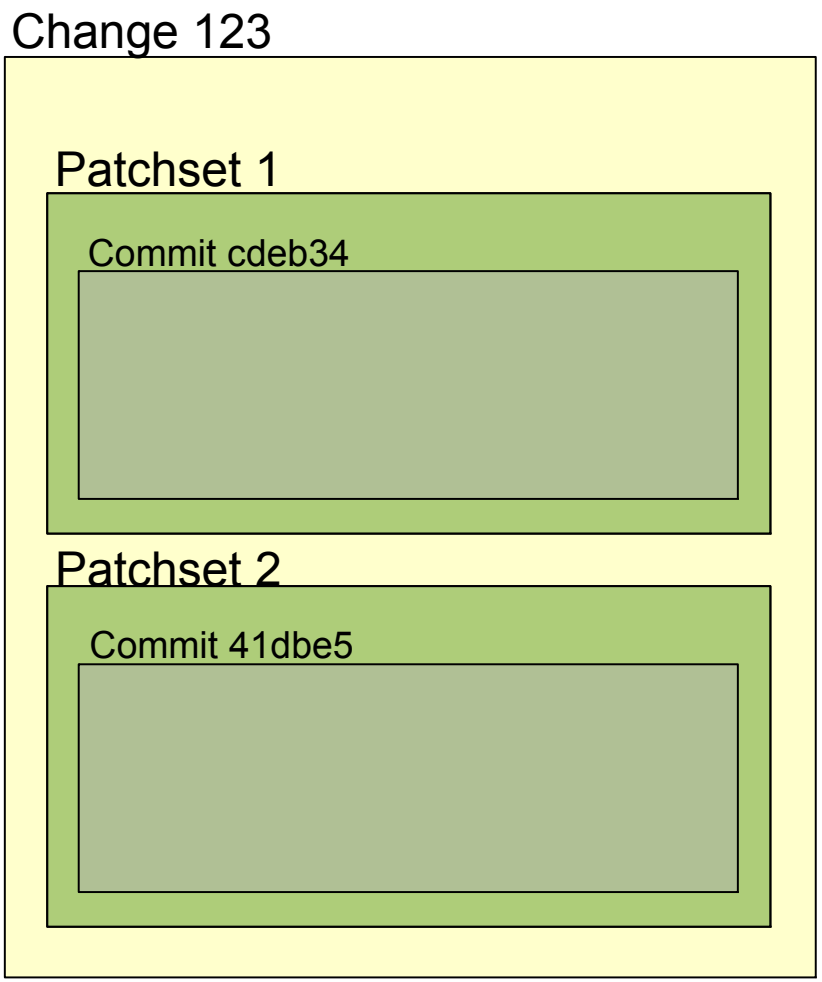
You never know how Google thinks of your patches exactly, even through Gerrit (code review system).

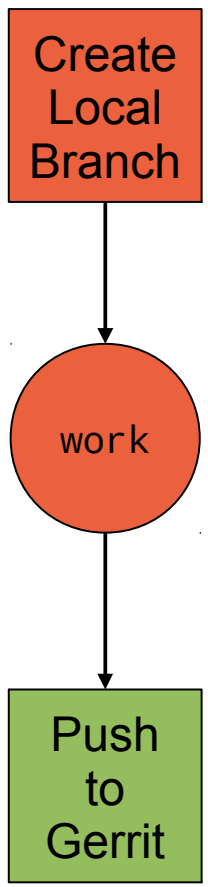
Problems We faced

- AOSP looks like “An Open Source Pretender"
- No public roadmap
  - **Therefore, we ignore the modifications against Android framework.**
- The merged changes usually show up in next 1 or 2 public release.
  - **It is really hard to introduce/track the relevant changes.**
- Not clear discussions on android-contrib mailing-list. Sometimes, you have to have private communications to Google engineers.
- Version control / Code Review on invisible repositories (internal and far-away GIT tree)

Change represented in Gerrit

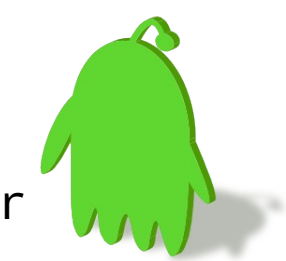






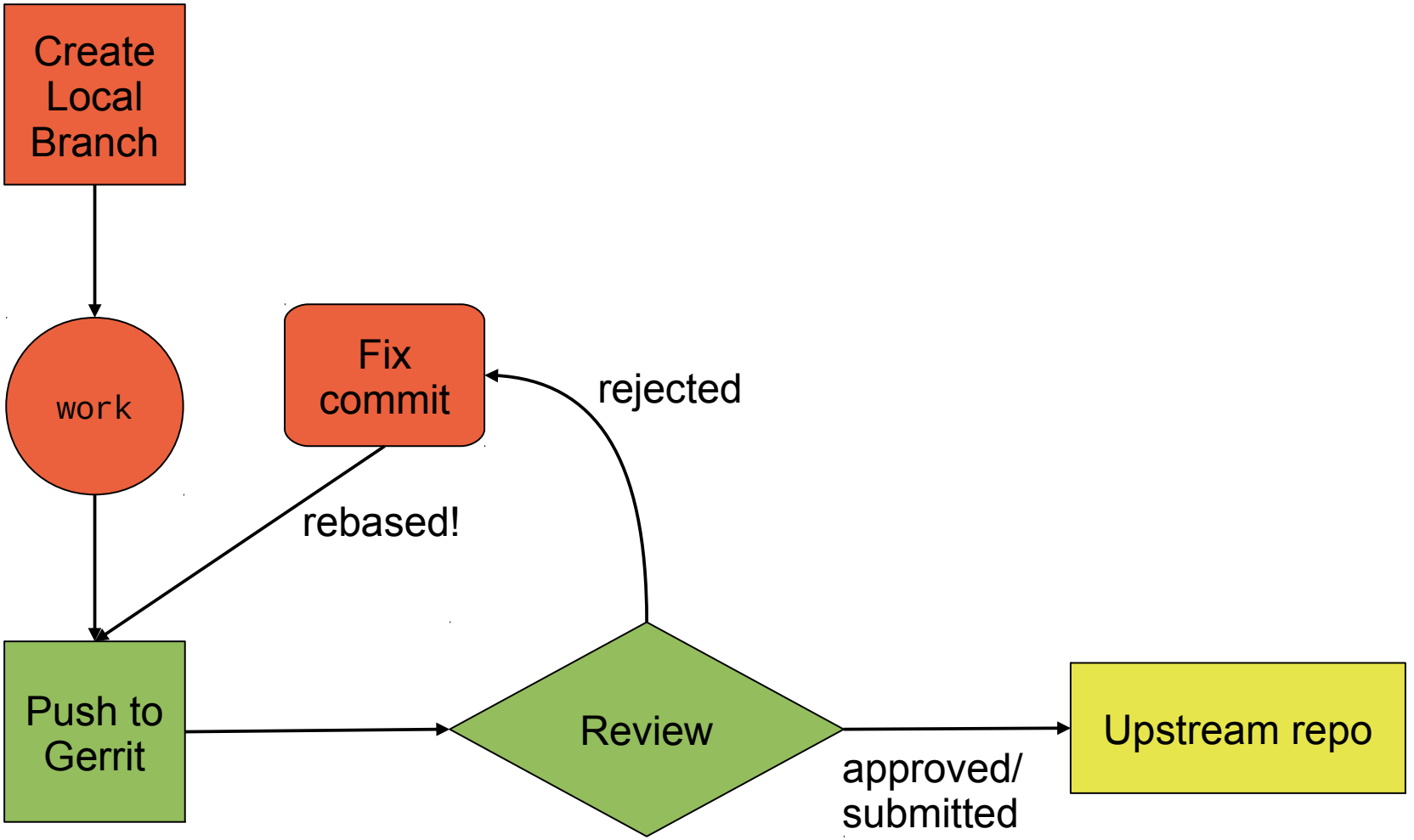

```
% git checkout -b topic-branch
```

```
% git push gerrit \
  HEAD:refs/for/master
```



Google provides the great code review tool for AOSP, but...

Flow of AOSP submitted changes



Here “upstream” means AOSP master and Google internal tree.

Applications Places System 日 8月 15, 11:10 上午

gitk: gcc

File Edit View Help

master — remotes/korg/master — remotes/m/master Remove definition of getpagesi
Backport upstream arm-linux-androideabi target to gcc-4.4.3.
commit gcc-4.4.3 which is used to build gcc-4.4.3 Android toolchain in master.
New gcc sources for toolchain checked in on 06/03/2010.
Merge "Add GNU-stack annotation to Android code."
Add GNU-stack annotation to Android code.
Fix a few divergence bugs and security problem.
gcc 4.2.1 compiler build for IA Android target
Backport upstream patch r152443.
Bring gcc-4.4.0 to up-to-date.
Gcc source for prebuilt toolchains in Eclair-mr2.
remotes/korg/eclair — remotes/m/eclair Check in gcc sources for prebuilt toolchains
Empty initial commit.

Jing Yu <jingyu@google.com> 2010-08-11 01:08:15
Jing Yu <jingyu@google.com> 2010-07-23 05:39:44
Jing Yu <jingyu@google.com> 2010-07-23 05:03:48
Jing Yu <jingyu@google.com> 2010-06-09 05:25:17
David Turner <dig
Jing Yu <jingyu@
Jing Yu <jingyu@
Bruce Beare <bru
Jing Yu <jingyu@
Jing Yu <jingyu@google.com> 2010-01-19 05:38:25
Jing Yu <jingyu@google.com> 2009-11-19 09:41:15
Jing Yu <jingyu@google.com> 2009-11-06 17:11:04
Jean-Baptiste Queru <jbq@

No explicit change history

Android style open source model

SHA1 ID: b094d6c4bf572654a031ecc4afe675154c886dc5 Row 3 / 13

Find next prev commit containing: Exact All fields

Search

Diff Old version New version Lines of context: 3 Ignore space change

```
Author: Jing Yu <jingyu@google.com>  2010-07-23 05:03:48
Committer: Jing Yu <jingyu@google.com>  2010-07-23 05:03:48
Parent: dc34721ac3bf7e3c406fba8cfe9d139393345ec5 (New gcc sources for toolchain checked i
Child:  f01e65905acd1e2052e875130cb723a658765bc8 (Backport upstream arm-linux-androideabi
Branches: master, migration, remotes/korg/master, remotes/m/master
Follows:
Precedes:

    commit gcc-4.4.3 which is used to build gcc-4.4.3 Android toolchain in master
    The source is based on fsf gcc-4.4.3 and contains local patches which
    are recorded in gcc-4.4.3/README.google.

    Change-Id: Id8c6d6927df274ae9749196a1cc24dbd9abc9887

---------------------------- gcc-4.4.3/ABOUT-NLS ----------------------------
new file mode 100644
index 0000000..47d5e39
@@ -0,0 +1,625 @@
+Notes on the Free Translation Pro
+**********************
+
```

Patch Tree

Comments
gcc-4.4.3/ABOUT-NLS
gcc-4.4.3/COPYING
gcc-4.4.3/COPYING.LIB
gcc-4.4.3/COPYING.RUNTIME
gcc-4.4.3/COPYING3

Gap between Goolge internal GIT/Perforce and AOSP Git

projects / toolchain/gcc.git / tree

summary | shortlog | log | commit | commitdiff | tree | review
snapshot

Remove definition of getpagesize() in order to build c++ library

[toolchain/gcc.git] /

drwxr-xr-x	gcc-4.2.1	tree | history
drwxr-xr-x	gcc-4.3.1	tree | history
drwxr-xr-x	gcc-4.4.0	tree | history
drwxr-xr-x	gcc-4.4.3	tree | history

After Gingerbread, it gets much clear for toolchain part

```
prebuilt/
commit 81cce608ab19dcd0aaf7d08d57a4460229e43c45
Author: Jing Yu <jingyu@google.com>
Date:   Tue Dec 14 10:55:23 2010 -0800

    Patched toolchain to fix a few gcc and binutils bugs.

    Sources to build this toolchain are listed on arm-eabi-4.4.3/SOURCES
```

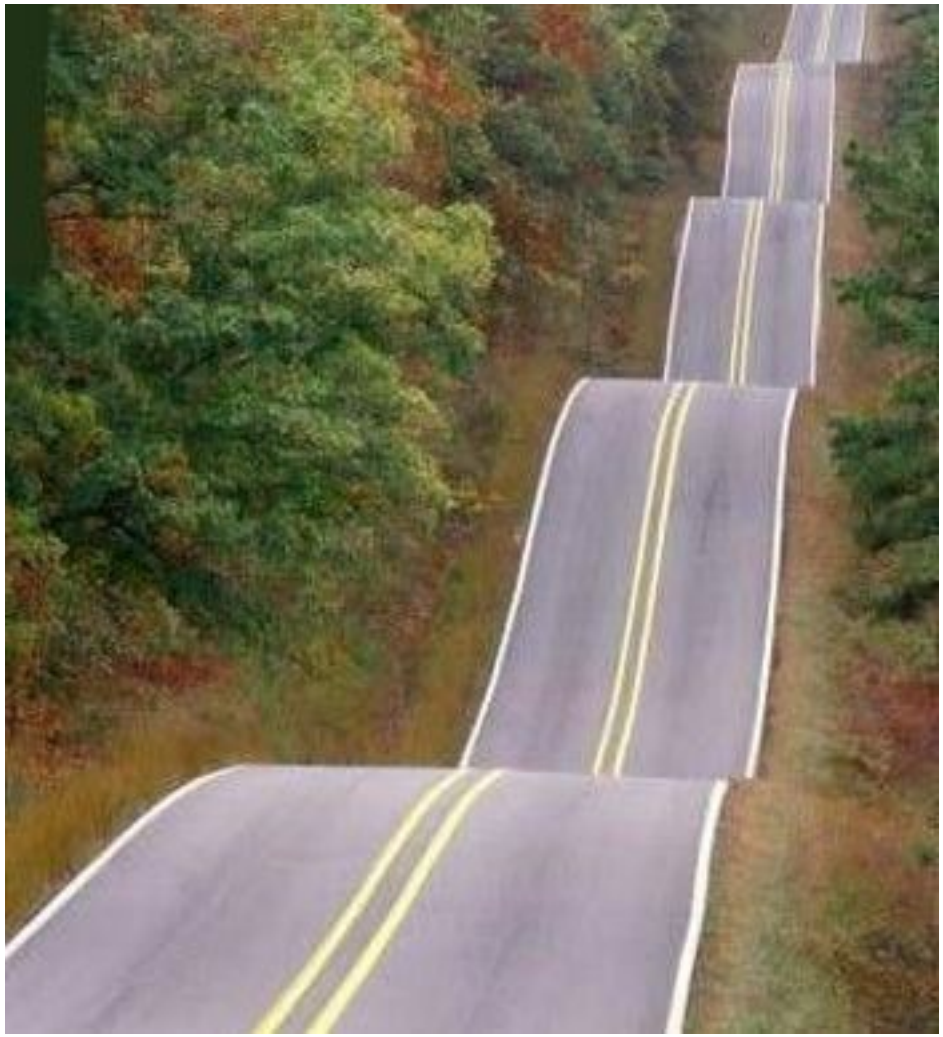
```
linux-x86/toolchain/arm-eabi-4.4.3/SOURCES

build/ synced to
        commit 4cc02faaa7e8828f9458b1828a6f85e7791ae2aa
        Author: Jim Huang <jserv@0xlab.org>
        Date:   Fri Aug 20 23:30:37 2010 +0800

   And rollback the following 3 patches.
   commit de263c26a7680529baca731c003bc58b68d72511
   Author: Jing Yu <jingyu@google.com>
   Date:   Thu Aug 12 15:52
```

Although we can check git log, we still have no idea why they changed

Observed AOSP Working Model

- “master” branch in Android

  is the bridge between AOSP and Google internal tree. There are many contributions merged from companies, organizations, and individuals. But no efficient code review available for non-existing repositories. And, only few Google engineers do review changes.
- Master branch = the latest AOSP + Partial changes by Google (bug-fixes from internal tree)
  - **Not fully verified codebase.**
- The best hints are the opinions written by Google engineers inside Gerrit.
  - **Send patches if possible**



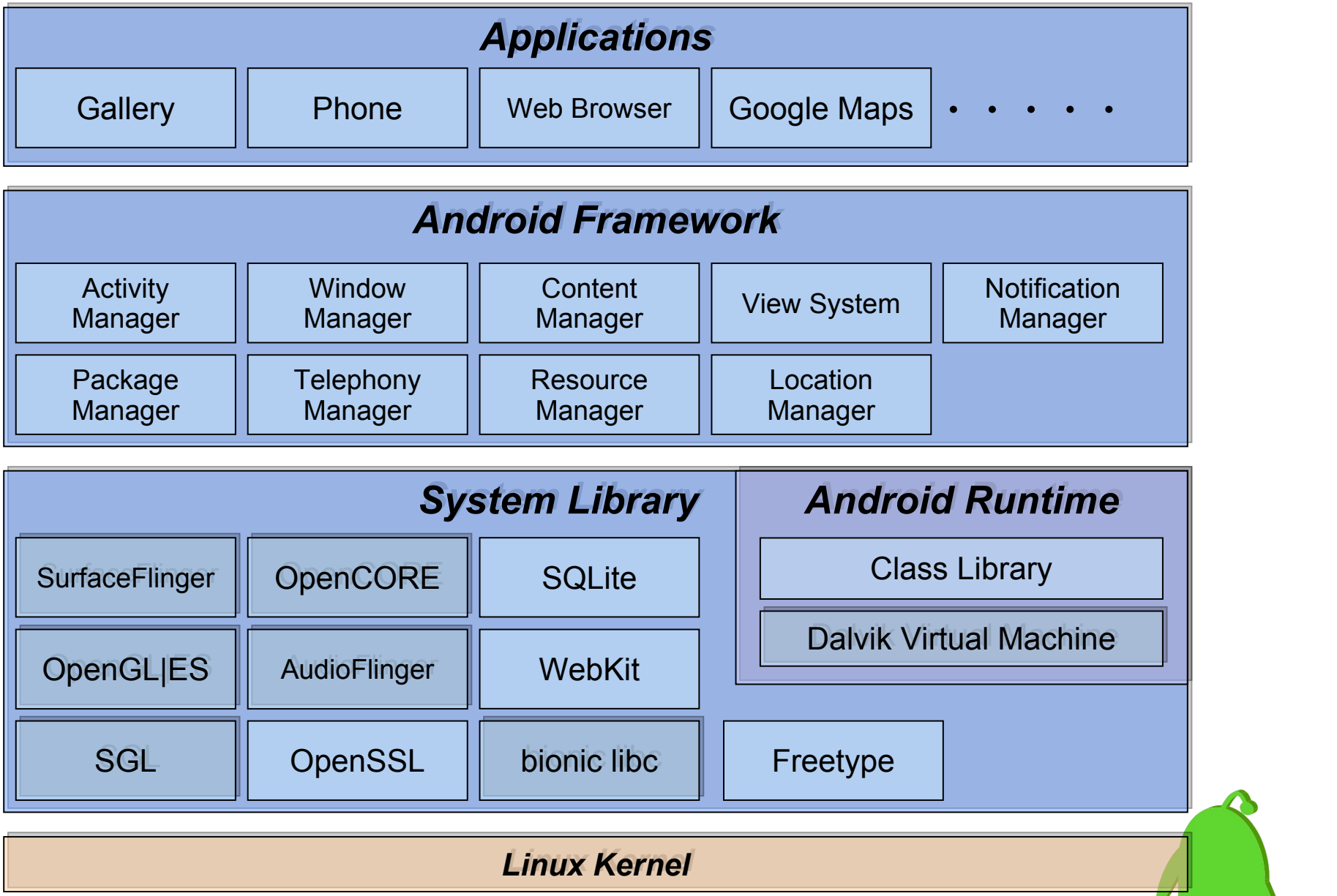
My interpretation of Android:

Hardware is Revolution;

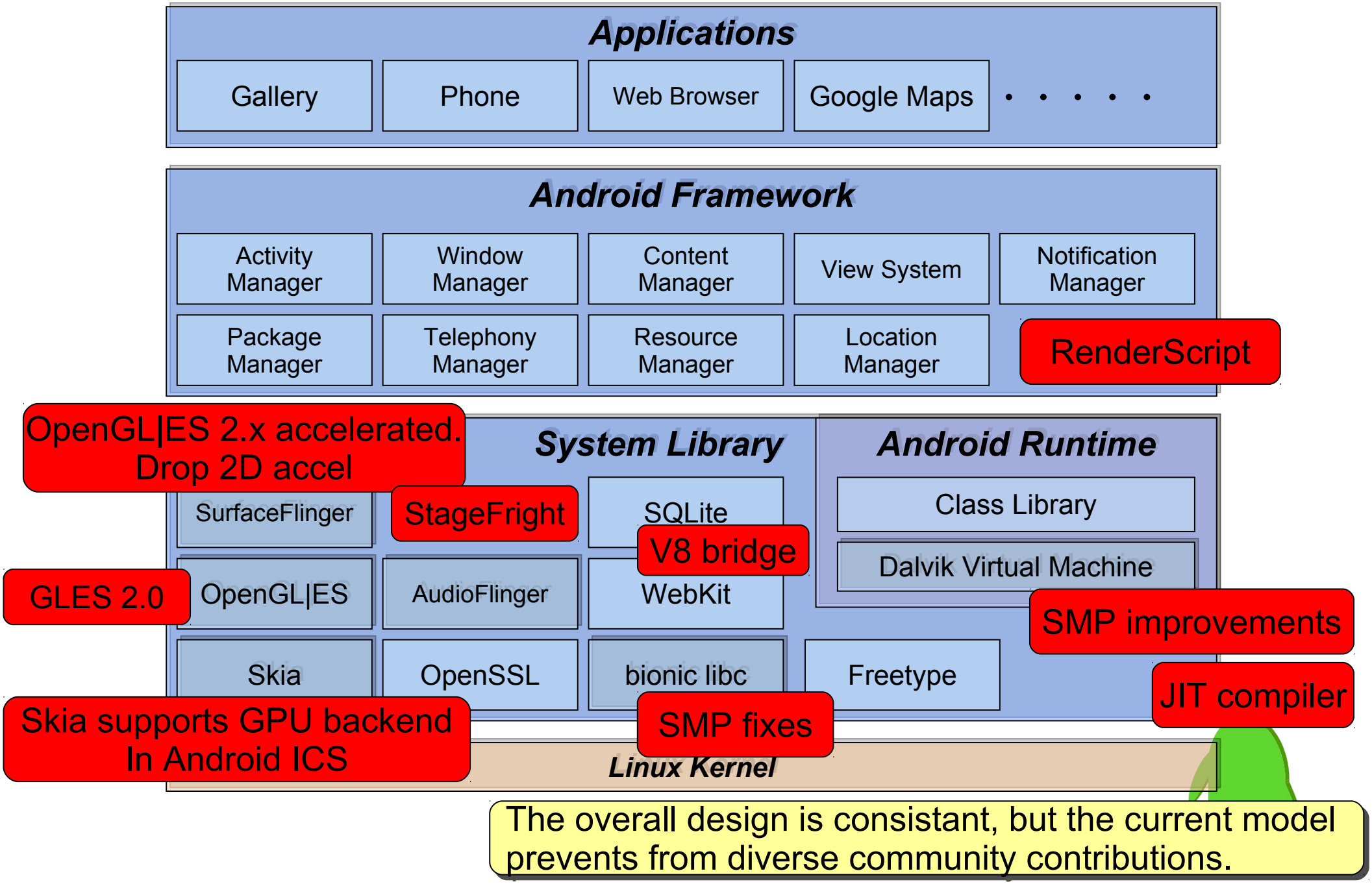
Sotware is basically Evolution;

Android is Hardware-driven Software Revolution

Functional View (Android 1.5)



Functional View (Android 2.3)



AOSP statistics for Gingerbread (Dec 2010)

- 4204 google.com
- 1354 android.com
- 98 sonyericsson.com
- 71 gmail.com
- 39 codeaurora.org
- 39 samsung.com
- 38 intel.com
- 32 nokia.com
- 32 holtmann.org
- 29 **0xlab.org**
- 25 trusted-logic.com
- 17 openbossa.org
- 11 nxp.com
- 11 linux.org.tw
- 10 ti.com
- 10 acer.com.tw
- 8 themaw.net
- 8 garmin.com
- 7 snpe.rs
- 7 motorola.com
- 7 mc.pp.se
- 7 googlemail.com

The number are commits since Froyo release.
However, the valuable changes from community such as CyanogenMod are usually absent due to long-time review process.

Ensure the Quality

when building custom Android distribution and merging changes from community

Mission in our development:

Improve UX in SoC

UX = User Experience

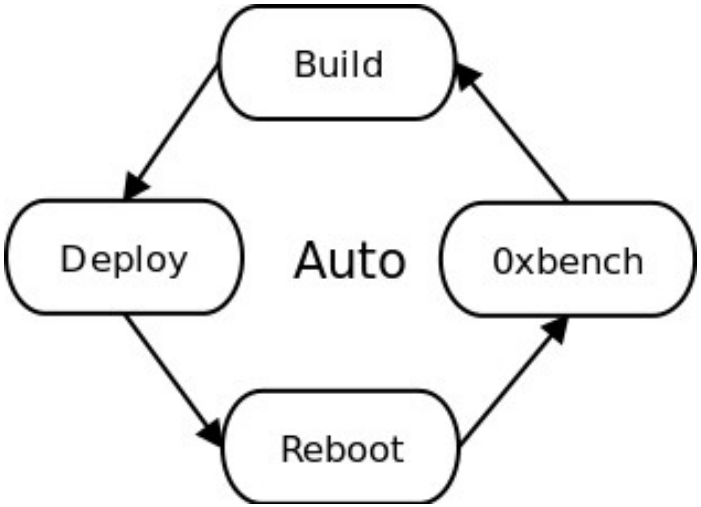
SoC = Integrated Computing Anywhere

Quality in custom Android Distribution

- 0xlab delivers the advantages of open source software and development.
  - **Quality relies on two factors: continuous development + strong user feedback**
- Several utilities are developed to ensure the quality and released as open source software.
  - **0xbench** (Android benchmarking tool)
  - **ASTER** (Android System Testing Environment and Runtime)
  - **LAVA** (Linaro Automated Validation Architecture)
- In the meanwhile, performance is improved by several patches against essential components.

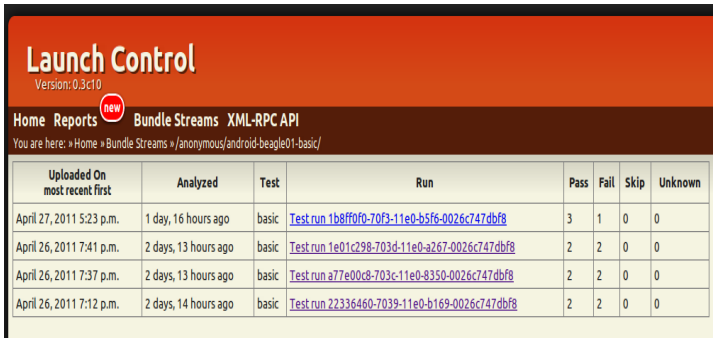
LAVA: Automated Validation Infrastructure for Android

Android benchmark running on **LAVA**. Automated Validation flow includes from deploy, then reboot, testing, benchmark running, and result submit.



Android support on LAVA
https://wiki.linaro.org/Platform/Validation/LAVA

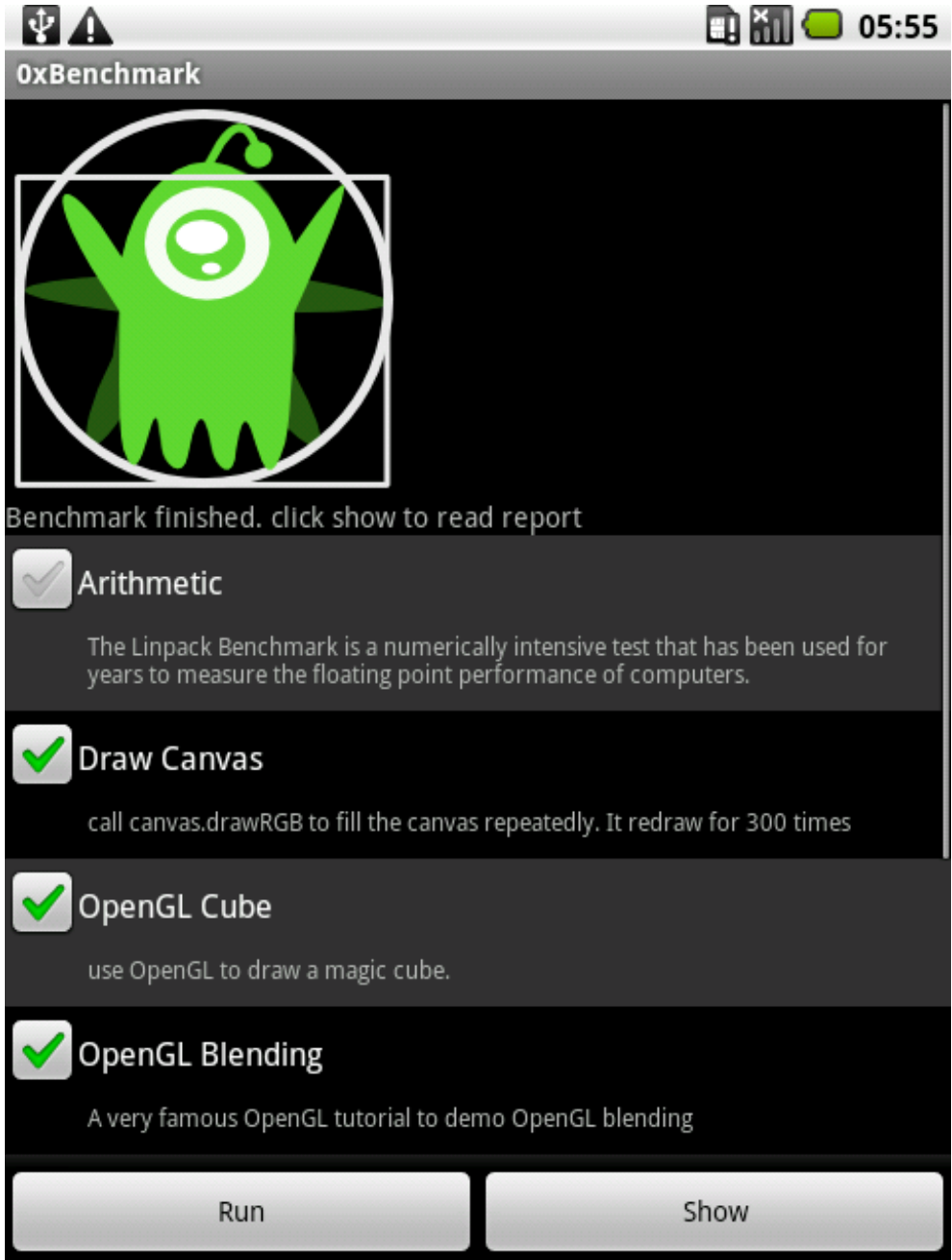
```
Android related commands in LAVA:
 * deploy_linaro_android_image
 * boot_linaro_android_image
 * test_android_basic
 * test_android_monkey
 * test_android_0xbench
 * submit_results_on_host
```

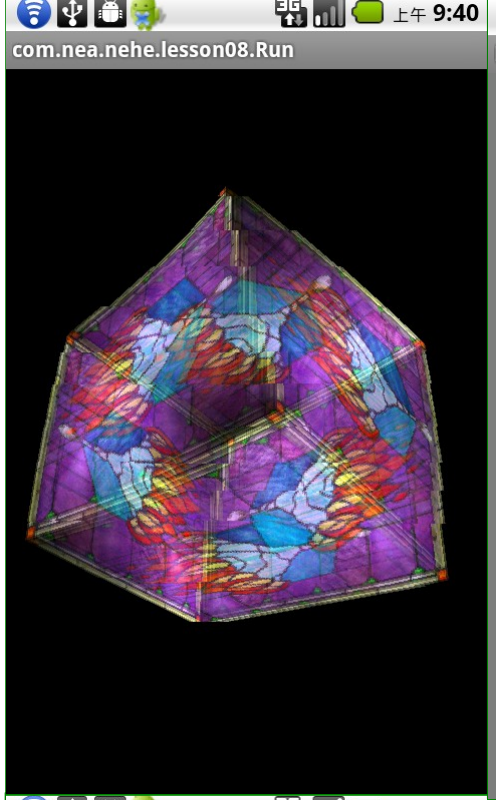


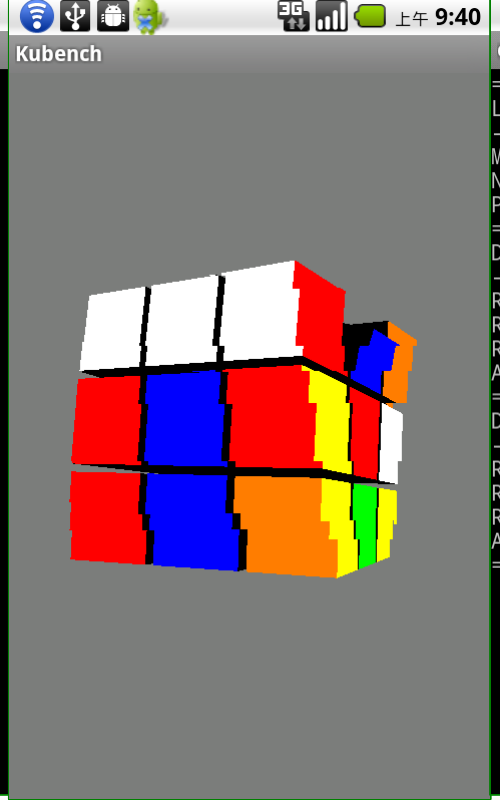
Uploaded On most recent first	Analyzed	Test	Run	Pass	Fail	Skip	Unknown
April 27, 2011 5:23 p.m.	1 day, 16 hours ago	basic	Test run 1b8ff0f0-70f3-11e0-b5f6-0026c747dbf8	3	1	0	0
April 26, 2011 7:41 p.m.	2 days, 13 hours ago	basic	Test run 1e01c298-703d-11e0-a267-0026c747dbf8	2	2	0	0
April 26, 2011 7:37 p.m.	2 days, 13 hours ago	basic	Test run a77e00c8-703c-11e0-8350-0026c747dbf8	2	2	0	0
April 26, 2011 7:12 p.m.	2 days, 14 hours ago	basic	Test run 22336460-7039-11e0-b169-0026c747dbf8	2	2	0	0

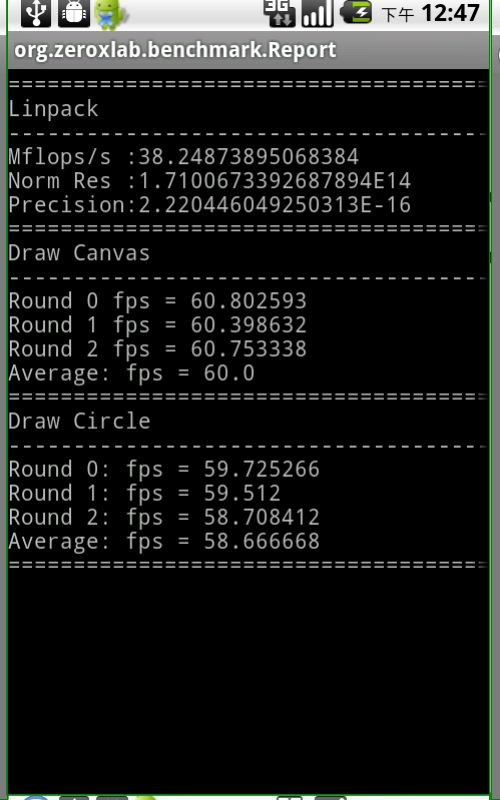
0xbench: comprehensive open source benchmark suite for Android

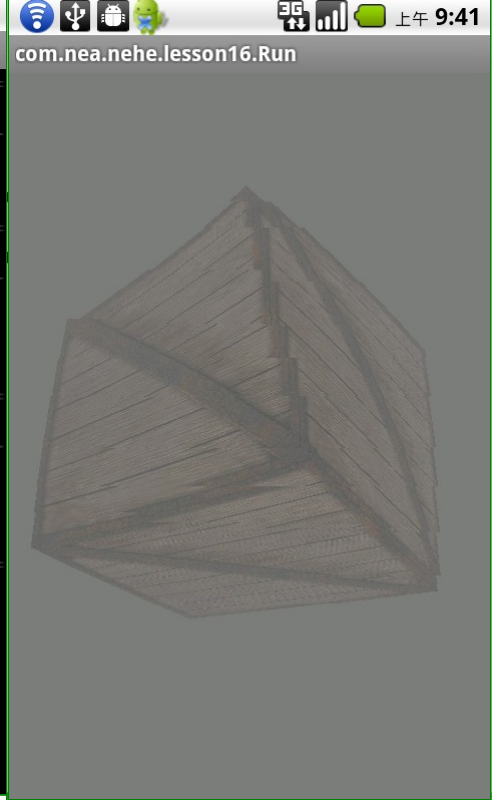
- A set of system utilities for Android to perform comprehensive system benchmarking
  - Dalvik VM performance
  - OpenGL|ES performance
  - Android Graphics framework performance
  - I/O performance
  - JavaScript engine performance
  - Connectivity performance
  - Micro-benchmark: stanard C library, system call, latency, Java invocation, ...

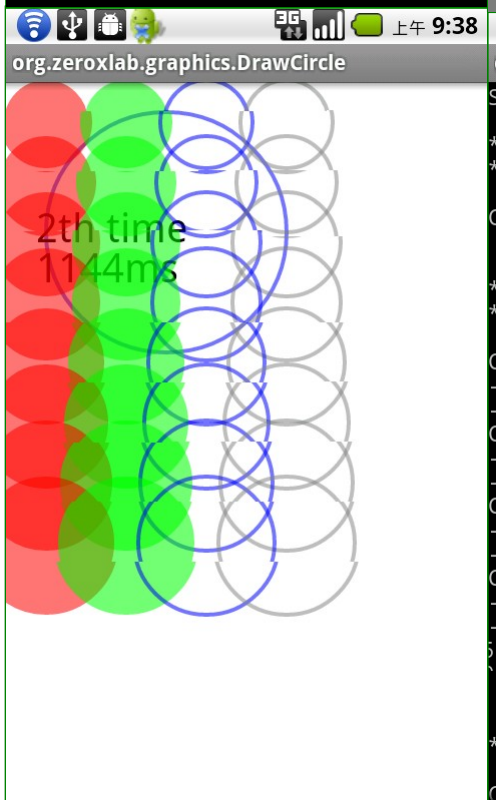



上午 9:40
com.nea.nehe.lesson08.Run

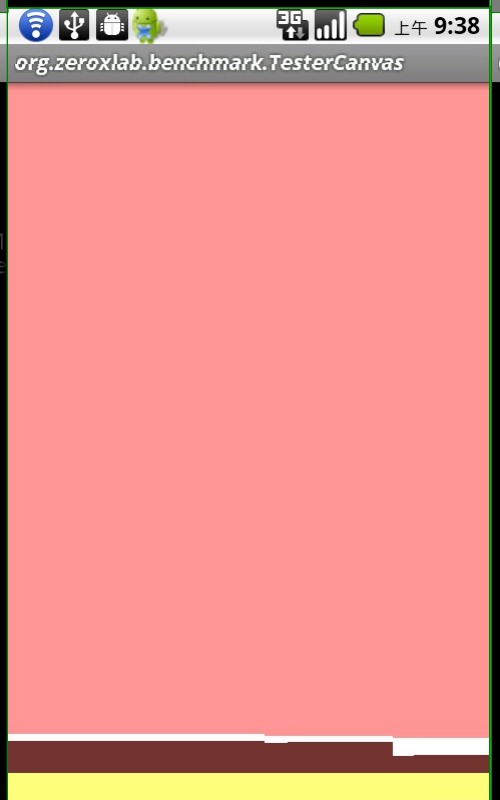

上午 9:40
Kubench


下午 12:47
org.zeroxlab.benchmark.Report
Linpack
Mflops/s :38.24873895068384
Norm Res :1.7100673392687894E14
Precision:2.220446049250313E-16
Draw Canvas
Round 0 fps = 60.802593
Round 1 fps = 60.398632
Round 2 fps = 60.753338
Average: fps = 60.0
Draw Circle
Round 0: fps = 59.725266
Round 1: fps = 59.512
Round 2: fps = 58.708412
Average: fps = 58.666668


上午 9:41
com.nea.nehe.lesson16.Run

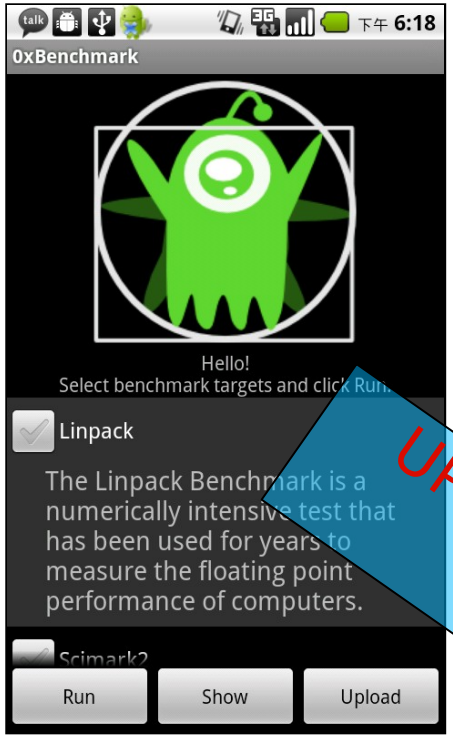

上午 9:38
org.zeroxlab.graphics.DrawCircle
2th time
1144ms


下午 12:46
org.zeroxlab.benchmark.TesterGC
Stretching memory:
binary tree of depth 16
*Total memory:3612640 bytes
*Free memory:738576 bytes
Creating:
long-lived binary tree of depth 1
long-lived array of 125000 double
*Total memory:8003552 bytes
*Free memory:3327960 bytes
Create 37448 trees of depth 2
- Top down: 1481msecs
- Bottom up: 1328msecs
Create 8456 trees of depth 4
- Top down: 1319msecs
- Bottom up: 996msecs
Create 2064 trees of depth 6
- Top down: 799msecs
- Bottom up: 776msecs
Create 512 trees of depth 8
- Top down: 957msecs
- Bottom up: 808msecs
*Free memory:2901808 bytes
Completed in 10413ms

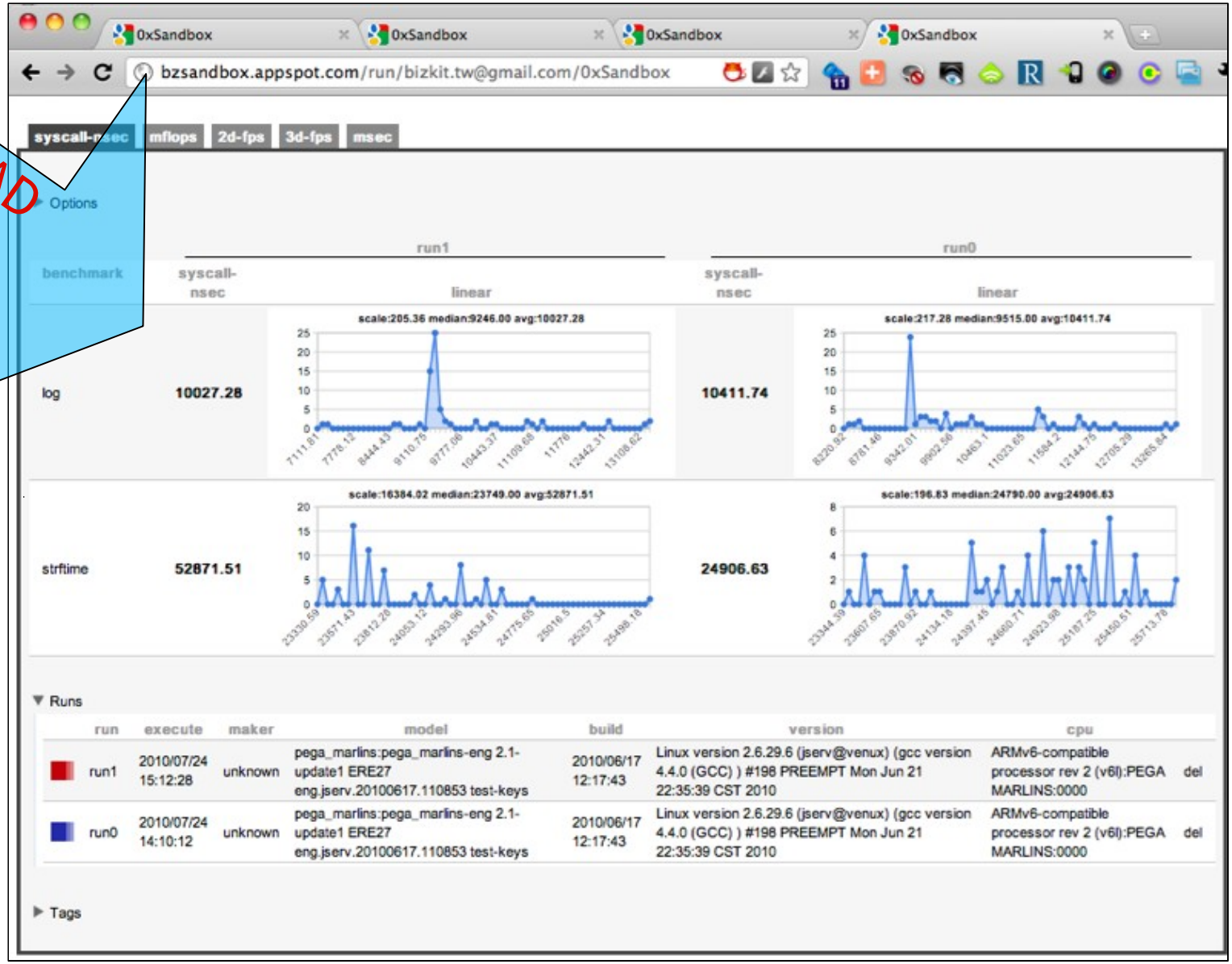

上午 9:38
org.zeroxlab.benchmark.TesterCanvas


上午 9:43
org.itri.teapot.TeapotES

Collect and Analyze results on server-side



UPLOAD



Android Functional Testing

(1) stress test

(2) Automated test

Stress Test

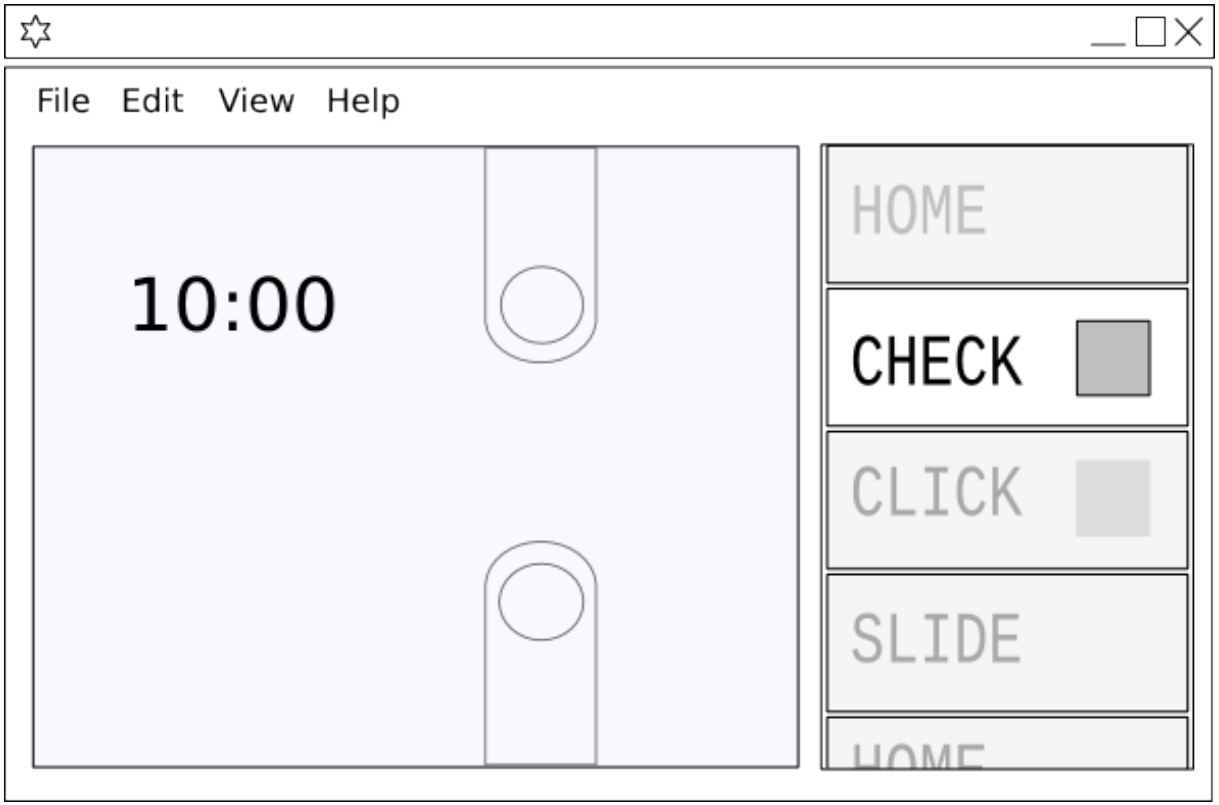
- According to CDD (Compatilbility Definition Document), Device implementations MUST include the Monkey framework, and make it available for applications to use.
- `monkey` is a command that can directly talks to Android framework and emulate random user input.

```
adb shell monkey -p your.package.name -v 500
```

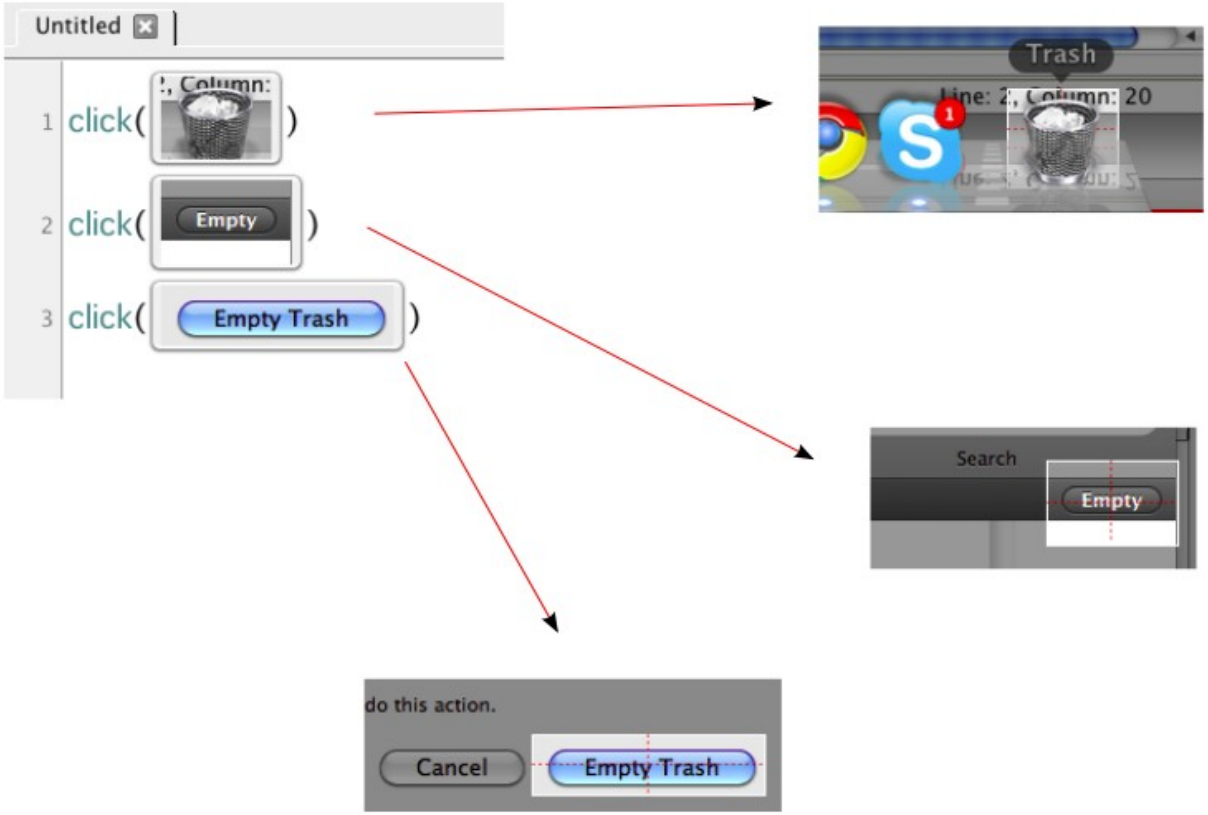
- Decide the percentage of touch events, keybord events, etc., then run automatically.

ASTER: Automated Test

- Blackbox-test vs. Whitebox-test
- An easy to use automated testing tool.



Desktop: Sikuli



Aster

Designed for non-programmer

Easy to use IDE

Batch executing of test scripts

Multiple chain of recall command

Aster


Aster
File Help
Recall
Touch
Angry Bird
Wait
Touch
Wait
PLAY
Done
Rotate
BETA
ANGRY BIRDS
LITE
PLAY
Play
Step
Stop

Bring Enhancements back to Community

What do we deliver to communty?

- Patches merged in AOSP, CyanogenMod, and Android-x86
- Implement 100% open source OpenGL|ES adaptation based on Mesa/3D into Android
  - **The world-first, important to Android-x86**
- Performance: ARM specific optimizations, graphics enhancement
- Features: Bluetooth HID (keyboard/mouse), external modem, 3D effects, customized Launcher

← run_Nexus_S:GRJ22_2011/07/13-22:48:03UTC

benchmark →	mflops	logarithmic
Linpack	14.83	
Scimark2:COMPOSITE	20.64	
Scimark2:FTT	13.43	
Scimark2:SOR	36.67	
Scimark2:MONTECARLO	5.72	
Scimark2:SPARSEMATMULT	18.37	
Scimark2:LU	28.99	

Arithmetic on Nexus S

Tune Dalvik VM performance (armv7)

run_Nexus_S:GRJ90_2011/08/03-18:11:16UTC

benchmark	mflops	logarithmic
Linpack	15.56	
Scimark2:COMPOSITE	21.84	
Scimark2:FTT	14.01	
Scimark2:SOR	38.53	
Scimark2:MONTECARLO	5.92	
Scimark2:SPARSEMATMULT	19.41	
Scimark2:LU	31.30	

2D on Nexus S

← run_Nexus_S:GRJ22_2011/07/13-22:48:03UTC

benchmark →	2d-fps	logarithmic
DrawCanvas	55.56	
DrawCircle	29.15	
DrawCircle2	51.23	
DrawRect	32.81	
DrawArc	47.12	
DrawImage	53.36	
DrawText	55.29	

Apply extra performance tweaks against optimized build (NEON)

← run_Nexus_S:GRJ90_2011/08/03-18:11:16UTC

benchmark →	2d-fps	logarithmic
DrawCanvas	56.06	
DrawCircle	33.19	
DrawCircle2	49.87	
DrawRect	42.42	
DrawArc	54.64	
DrawImage	55.85	
DrawText	55.44	

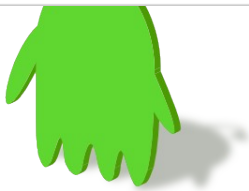
Benchmark: 2D (arm11-custom)

mflops | 2d-fps | 3d-fps | msec

▶ Options

benchmark	advanced-performance2 2d-fps	advanced-performance2 linear	advanced-performance 2d-fps	advanced-performance linear	startpoint 2d-fps	startpoint linear
DrawCanvas	49.93		48.38		14.65	
DrawCircle	23.29		22.68		10.32	
DrawCircle2	18.84		18.80		9.77	
DrawRect	7.64		8.80		5.76	
DrawArc	14.92		14.32		8.40	
DrawImage	5.59		5.50		3.10	
DrawText	19.56		19.44		9.00	

benchmark	M3 + Linaro Toolchain 2d-fps	M3 + Linaro Toolchain linear	M3 2d-fps	M3 linear	2.6.35 (2.6.32 pmem) 2d-fps	2.6.35 (2.6.32 pmem) linear
DrawCanvas	58.35		58.57		38.64	
DrawCircle	38.91		37.53		22.32	
DrawCircle2	18.67		17.92		19.64	
DrawRect	19.71		19.26		16.23	
DrawArc	26.84		24.68		24.66	
DrawImage	6.73		6.69		6.22	
DrawImage2	19.16		19.06		15.69	
DrawText	29.22		29.28		25.66	



Benchmark: 3D (arm11-custom; no GPU)

mflops | 2d-fps | 3d-fps | msec

▶ Options

benchmark	advanced-performance2 3d-fps	advanced-performance2 linear	advanced-performance 3d-fps	advanced-performance linear	startpoint 3d-fps	startpoint linear
OpenGLCube	27.65		26.36		11.77	
OpenGLBlending	15.21		15.06		8.78	
OpenGLFog	14.03		13.86		8.36	
FlyingTeapot	12.30		11.26		7.38	

benchmark →	← M3 + Linaro Toolchain 3d-fps	← M3 + Linaro Toolchain linear	← M3 3d-fps	← M3 linear
OpenGLCube	29.06		29.04	
OpenGLBlending	20.07		19.94	
OpenGLFog	18.63		18.95	
FlyingTeapot	17.49		17.04	

This explains that we have several system tools and development flow to help customers/community to verify the performance and improve.

Bionic libc

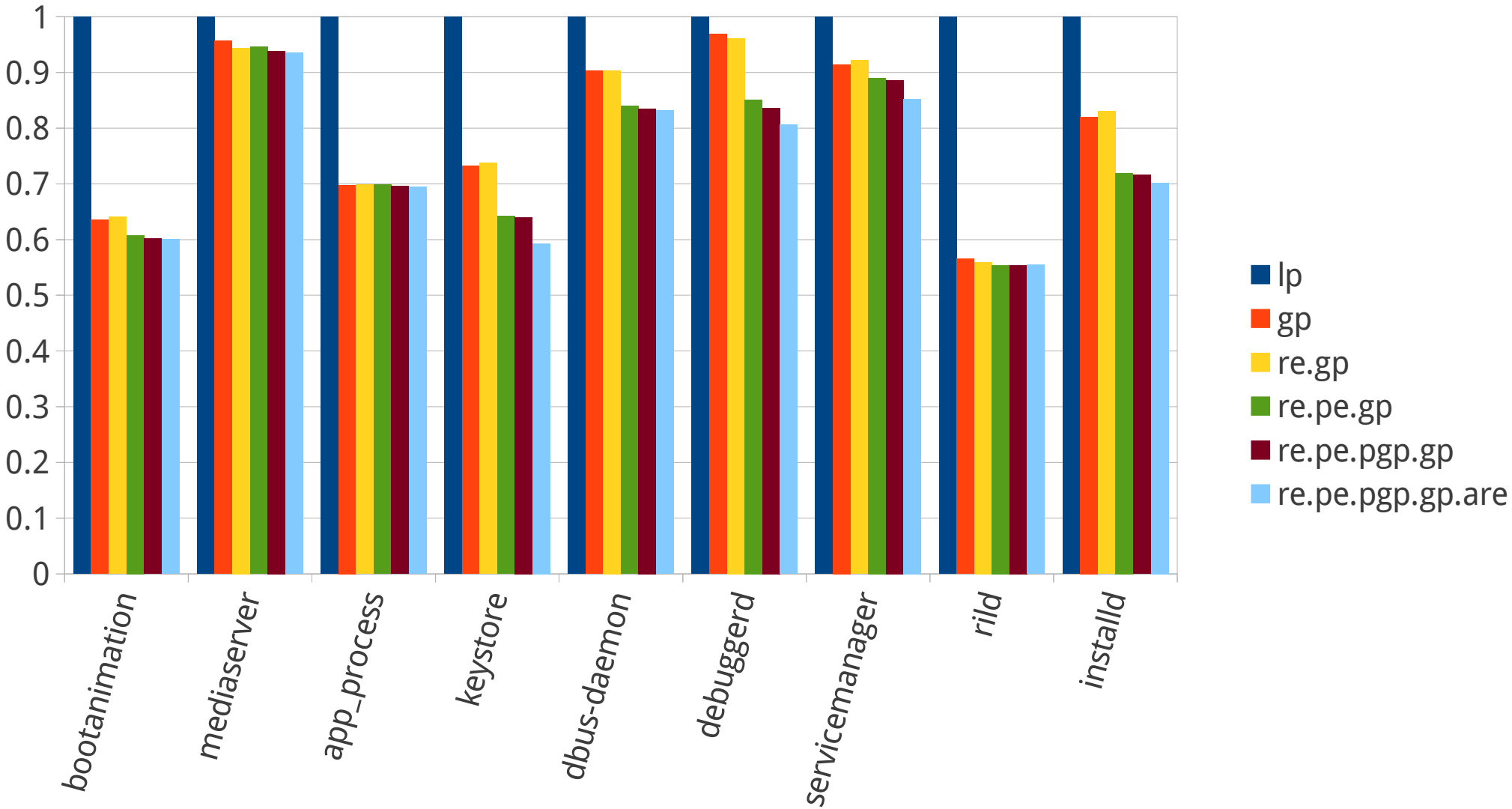
- Android C/C++ library
- 0xlab's Optimizations (merged in Android upstream)
  - Memory operations: Use ARMv6 unaligned access to optimize usual cases
  - Endian/Data Type conversion: Use ARMv6 fast endian primitives. Useful for TCP/IP (big endian / little endian converting)
  - Various ARM optimized string operations
    - memcpy, strcmp, strcpy, memset

Dynamic Linker Optimizations

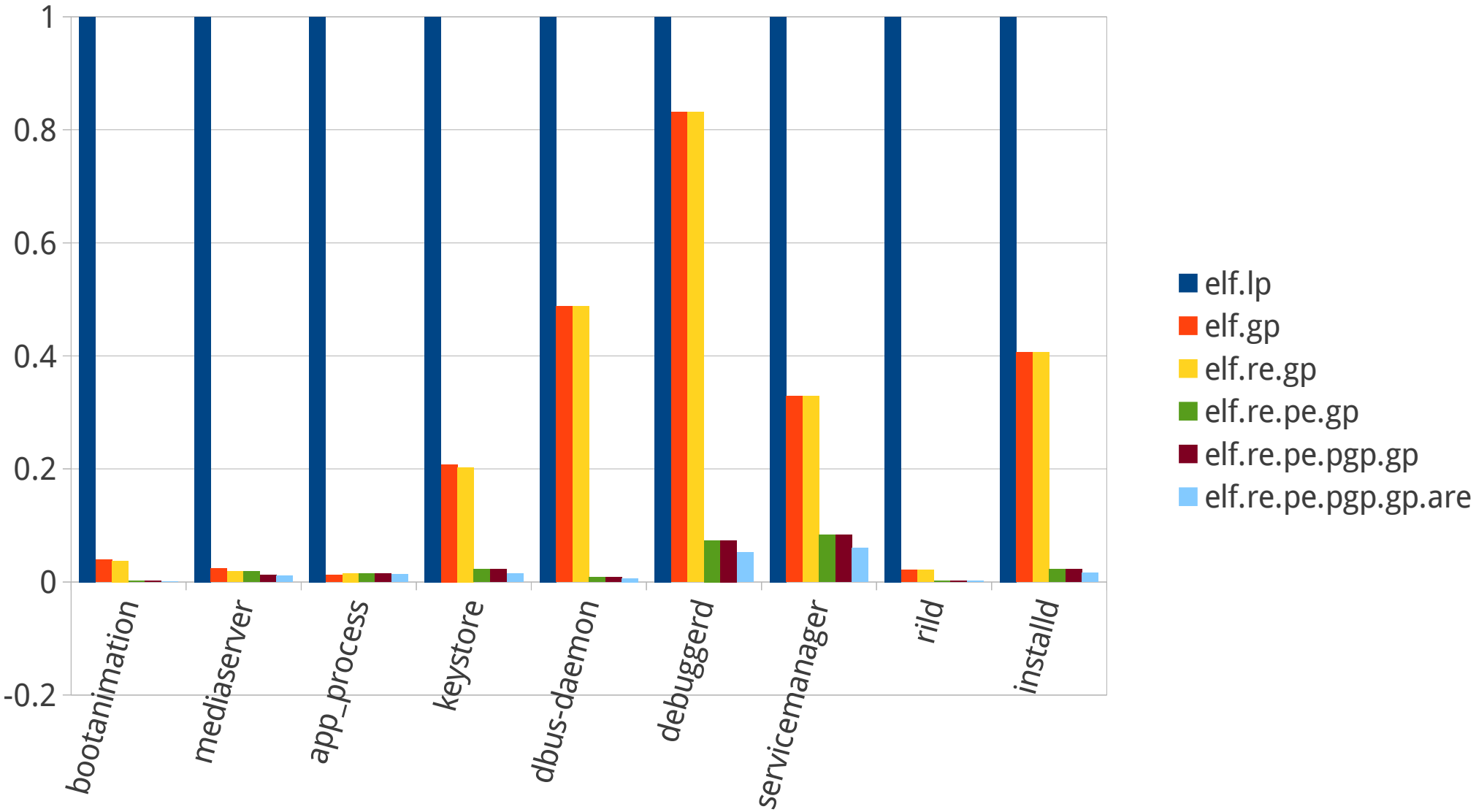
Why and How?

- The major reason to optimize dynamic linker is to speed up application startup time.
- Approaches:
  - **Implement GNU style hash support for bionic linker**
  - **Prelinker improvements: incremental global prelinking**
    - **reduce the number of ELF symbol lookup aggressively**
- Changed parts
  - **apriori, soslim, linker, elfcopy, elfutils**

(normalized) Dynamic Link time



(normalized) Symbol Lookup number

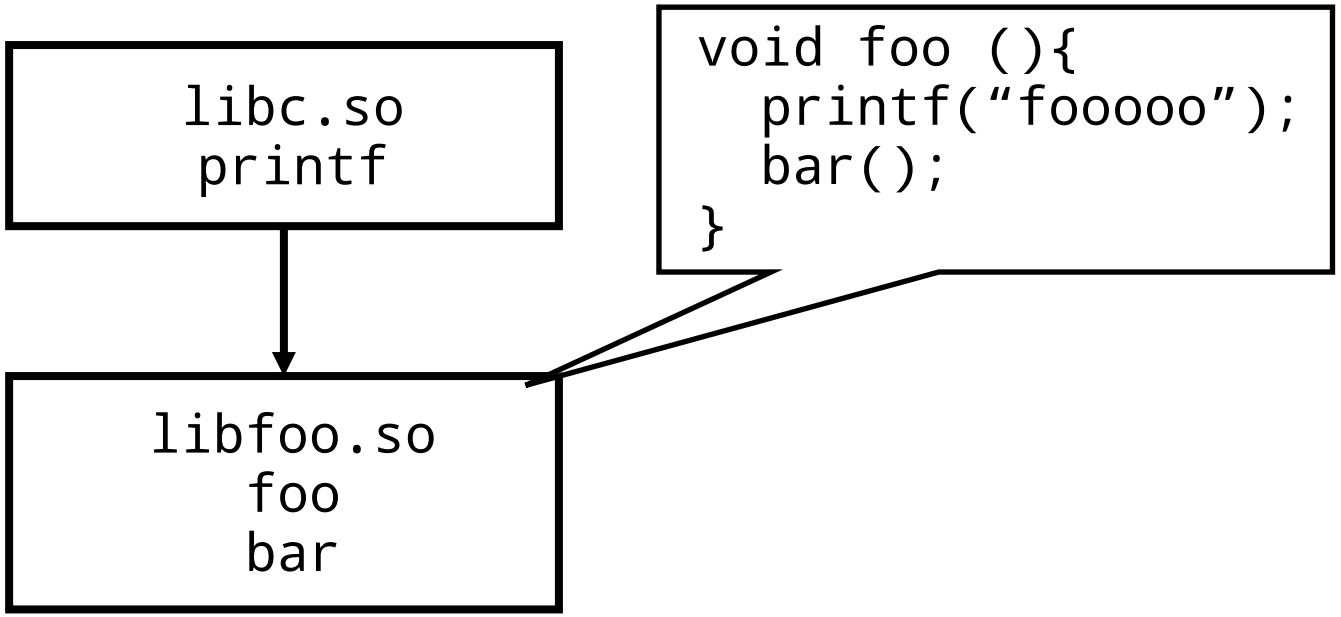


DT_GNU_HASH: visible dynamic linking improvement =

Better hash function (few collisions)

+ Drop unnecessary entry from hash

+ Bloom filter



libfoo.so

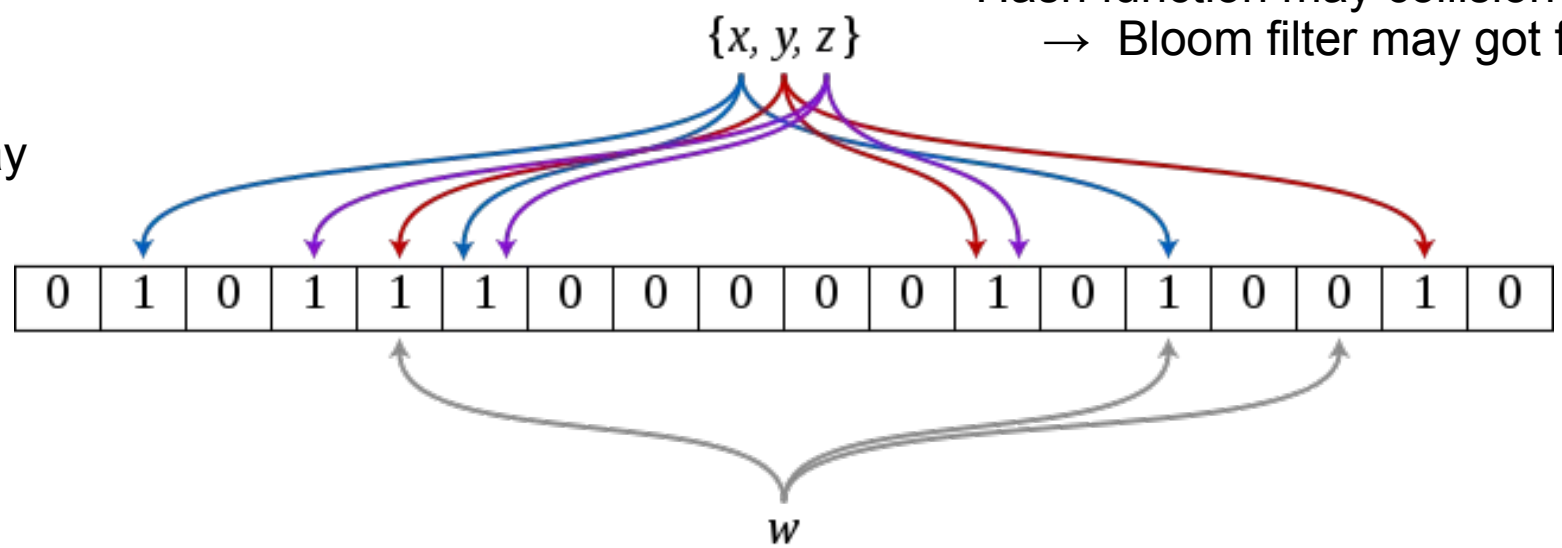
DT_GNU_HASH	DT_HASH
foo	foo
bar	bar
	printf

	Symbols in ELF	lookup#	fail#	gnu hash	filtered by bloom
gnu.gp	3758	23702	19950	23310	18234(78%)
gnu.gp.re	3758	20544	16792	19604	14752(75%)
gnu.lp	61750	460996	399252	450074	345032(76%)
gnu.lp.re	61750	481626	419882	448492	342378(76%)

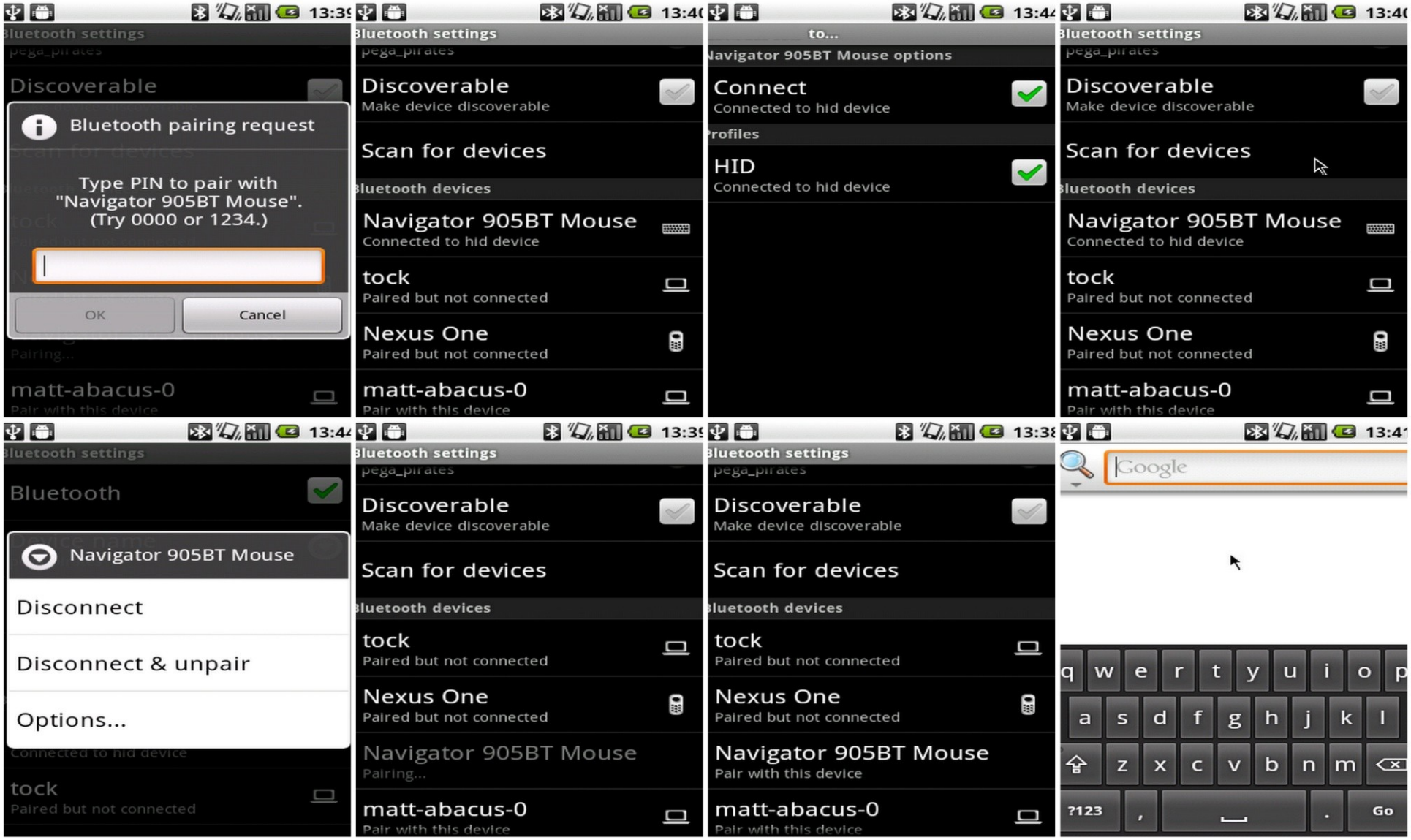
H = {x, y, z} = hash functions

Hash function may collision
→ Bloom filter may got false positives

Bit array



Bluetooth HID (Keyboard/Mouse)



UI customizations

- Provide several UI/Launcher combination for small and large screen devices.
  - Sizing from HVGA, VGA, SVGA (Phone), to 720p/1080p (TV)
- Either modified Android Launcher or new replacement
- Licensed under Apache Software License

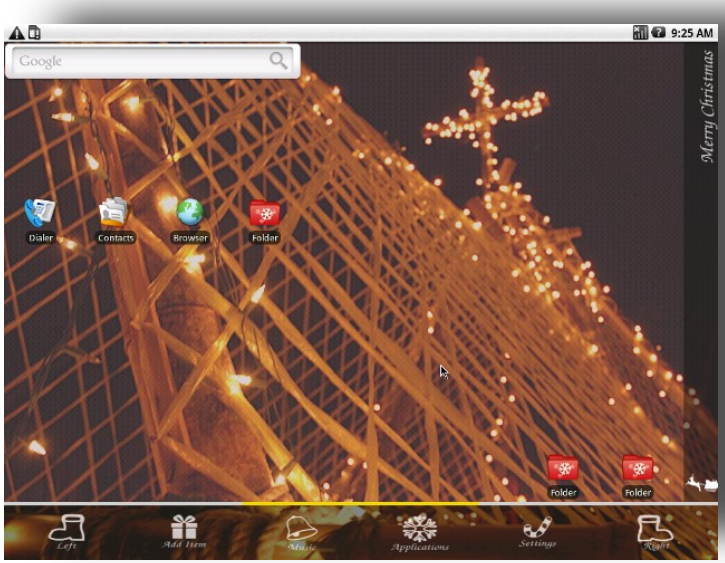
Some UI Changes

- Hardware enablement: Beagleboard (TI OMAP3),
- BottomBar

  Source code: **http://gitorious.org/0xdroid/packages_apps_launcher**
- PositionBar

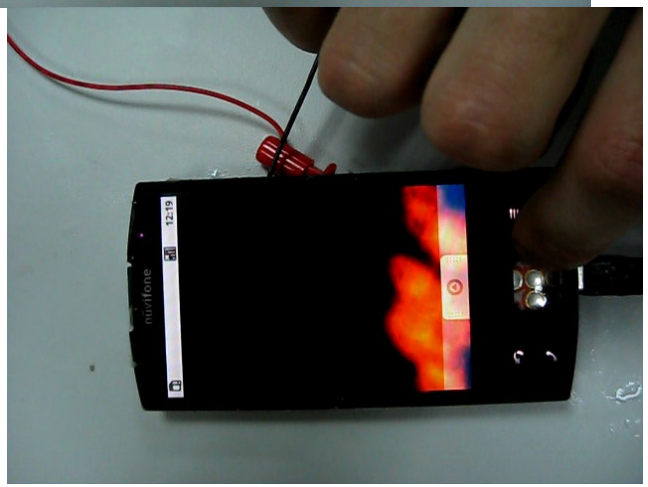
  Visible Hint
- ThemeSelector

  http://code.google.com/p/0xdroid/wiki/LauncherTheme



Products with Advanced 3D UI

- 3D effects and the ARM optimizations are enabled in Qualcomm platforms: MSM7x27 (with GPU) and MSM7x25 (software only)



Android Boot Time Optimizations

Boot loader

- Qi Boot-loader
  - **Only one stage boot-loader**
  - **Small footprint ~30K**
  - **Currently support**
    - **iMX31**
    - **Samsung 24xx**
    - **Beagleboard**
  - **KISS concept**
    - **Boot device and load kernel**

	Qi Boot-oader	U-Boot + XLoader
Size	~30K	~270K+20K
Time to Kernel	< 1 s	> 5s
Usage	Product	Engineering
Code	Simple	Complicated

Optimized ARM Hibernation

- Based on existing technologies thus requires little modification to userspace
  - **TuxOnIce**
- Release clean-pages before suspend
- Swap out dirty-pages before save image
- Image size reduced leads to faster resume time.

Further Boot Time Optimization

- Save the heap image (like core dump) of Zygote after preloading classes
- Modify Dalvik to make hibernation image after system init and before Launcher startup
- Parallize Android init
- Cache & Share JITed code fragment

Resources

- 0xdroid Roadmap: http://code.google.com/p/0xdroid/wiki/Roadmap
- Source repository: http://gitorious.org/+0xlab
- Wiki: http://code.google.com/p/0xdroid/w/list
- Demo videos: http://www.youtube.com/0xlab
- Mailing-list:
  - General discussion: http://groups.google.com/group/0xlab-discuss
  - Technical / Development: http://groups.google.com/group/0xlab-devel
- IRC channel (FreeNode): #0xlab

Thanks for Attending

Special thanks to AzureWave, who sponsors me for a long time.

AzureWave

Any Questions?

http://0xlab.org